ISSN 1864-6168
UDK 62

RAZVOJ JEDNOSTAVNOG ALATA ZA ANALIZU ZVUKA NA MOBILNOJ ANDROID PLATFORMI

DEVELOPMENT OF A SIMPLE TOOL FOR AUDIO ANALYSIS ON MOBILE ANDROID PLATFORM

Dražen Hižak, Matija Mikac

Stručni članak

Sažetak: *Svakim danom razvija se sve više aplikacija za „pametne“ mobilne uređaje bazirane na najpopularnijim mobilnim platformama – Android, iOS, Windows Phone i ostalima. Mobilni uređaji se sve učestalije primjenjuju i koriste za lakše obavljanje određenih zadaća. Kao takav priručni uređaj pokazuju sve veći tržišni potencijal. U ovom članku opisuje se tijek razvoja aplikacije za obradu zvuka na mobilnoj platformi zasnovanoj na operacijskom sustavu Android, namijenjene jednostavnoj analizi amplitudno-vremenske domene audiosignala. Programsko rješenje implementira tri jednostavne funkcije za obradu signala i vizualizaciju snimljenog uzorka. U članku se iznosi i kratak pregled izvedbe grafičko korisničkog sučelja bez kojeg ne bi bilo moguće upravljati ovom aplikacijom, te neke osnove digitalne obrade signala.*

Ključne riječi: *Android, pametni telefon, amplituda, audiosignal, digitalna obrada signala, zvuk*

Professional paper

Abstract: *Every day more applications for smartphone devices are being developed – on Android, iOS, Windows Phone, as well as other platforms. Mobile devices are being used for a rising number of everyday tasks, and as such convenient gadgets they are showing great market potential. This article is following the development of an application for sound editing on an Android mobile platform and deals with a simple analysis of an audio signal's amplitude-time domain. It implements three simple functions for signal processing and recorded sample visualisation. The graphic interface, without which the control of this application would not be possible, is also being discussed, along with some basics of digital sound analysis.*

Key words: *Android, smartphone, amplitude, audio signal, digital signal processing, sound*

1. UVOD

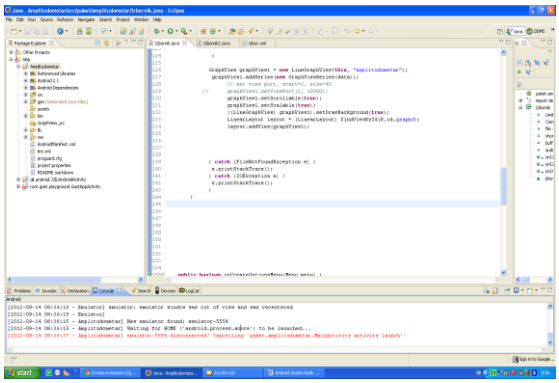
Mobilni uređaji se sve učestalije primjenjuju u svakodnevnom životu i koriste za lakše obavljanje određenih zadaća. Kao takav priručni uređaj pokazuju sve veći tržišni potencijal. Jedna od mogućih primjena je i obrada zvučnih zapisa.

Ovaj članak daje pregled tijeka razvoja aplikacije za jednostavnu obradu zvuka na mobilnoj platformi baziranoj na operacijskom sustavu *Android*. Opisuju se osnove digitalne obrade signala te konkretni koraci razvoja grafičkog korisničkog sučelja, modula za vizualizaciju signala i implementirane funkcije za obradu zvuka. Implementirane funkcionalnosti odnose se na obradu zvuka u vremenskoj domeni. Razvijena aplikacija za jednostavnu obradu zvuka koristi se prije svega u edukativne svrhe i dio je završnog rada studenta [1]. Osim što omogućava dohvat zvuka preko standardnog mikrofona na uređaju, aplikacija nudi mogućnost vizualizacije signala, obradu i modifikaciju signala ovisno o korisnički definiranim parametrima, te mogućnost preslušavanja signala prije ili nakon modifikacija.

2. ODABIR MOBILNE PLATFORME

Unatoč tome što postoji više mobilnih operacijskih sustava, *Android* mobilna platforma je odabrana, osim zbog zastupljenosti, prije svega zbog pristupačnosti alata za razvoj aplikacija kao i relativne jednostavnosti razvoja. Dok je za razvijanje aplikacija na mobilnoj platformi *iOS* potrebno imati računalo proizvođača *Apple* i plaćati godišnju pristojbu za pristup službenoj dokumentaciji, *Google* pruža besplatnu podršku i pristup dokumentaciji [2], a i programiranje samih aplikacija se može radi neovisno o operacijskom sustavu računala, pri čemu su razvojna okolina i dostupni alati besplatni.

Za razvoj aplikacije korištena je *Eclipse* razvojna okolina, (slika 1.) [3], za koju *Google* pruža direktnu podršku. Nakon preuzimanja razvojne okoline treba preuzeti paket alata ADT (eng. *Android Development Tools*) sa službenih stranica *Android* mobilne platforme. ADT paket sastoji se od emulatora mobilnih uređaja, programa za povezivanje razvojne okoline s mobilnim uređajem, program za praćenje izvođenja aplikacija i drugih, u razvoju korisnih, pomoćnih alata.



Slika 1. Razvojno okruženje *Eclipse*

Verzija *Android* sustava na kojem je razvijana aplikacija je 2.3 (API *level* 10). Pri odabiru razvojne verzije sustava bitno je da je aplikaciju moguće izvesti sa što nižom razinom API-ja kako bi ju bilo moguće pokrenuti na što više mobilnih uređaja. S obzirom na udio u zastupljenosti na *Android* mobilnim uređajima (nastanak softvera u rujnu 2012.), kao optimalni izbor odabrana je verzija 2.3.

Još jedan bitan faktor je i odabir mobilnog uređaja. Da bi se signal mogao kvalitetno obraditi u što kraćem vremenu (što brže), potreban je i mobilni uređaj veće procesorske snage. Gotovo svi danas dostupni modeli uređaja baziranih na *Android* operacijskom sustavu zadovoljavaju što se tiče izvedbe implementiranih funkcionalnosti, pri čemu bi većina od njih osigurala i mogućnost izvedbe složenijih funkcija.

3. OSNOVE DIGITALNE OBRADE SIGNALA

3.1. Svojstva analognih signala

Zvuk je mehanički val uzrokovan vibracijama nekog objekta (gitara, zvučnik, okolina). Ono što je bitno kod tih vibracija za nas je čujno područje koje se proteže od približno 16 Hz do 20 kHz. Da bi mogli baratati zvukom potrebno ga je prvo pretvoriti u električni signal, a to se postiže mikrofonom. Svojstvo svakog električnog signala dohvaćenog mikrofonom je da ima određenu frekvenciju i određenu amplitudu (napon). Da bi se taj signal mogao obrađivati, postoje dvije opcije. Prva opcija je da taj signal bude prvo snimljen na neki medij za pohranu podataka, a onda obrađen, dok je druga opcija da ga se obrađuje u stvarnom vremenu uz određeni vremenski odmak koji se nastoji minimizirati.

Vezano uz obradu signala na računalu, prije same obrade nužno je provesti digitalizaciju signala. Postupak uključuje nekoliko koraka: uzimanje uzoraka (uzorkovanje), kvantizaciju i kodiranje.

3.2. Uzorkovanje

Da bi neki kontinuirani (analogni) signal mogli obrađivati (analizirati na računalu ili, u našem slučaju, "pametnom" mobilnom telefonu), potrebno je signal digitalizirati - dohvatiti određeni broj uzoraka signala (vrijednosti) u određenom vremenskom periodu. Broj uzoraka koje uzimamo u sekundi određuje se tzv. frekvencijom uzorkovanja. Da bi se uzorkovani signal mogao pravilno rekonstruirati, frekvencija uzorkovanja mora biti barem dvostruko veća od maksimalne frekvencije signala. To opisuje Nyquistov teorem, pri čemu se frekvencija uzorkovanja naziva i Nyquistovom frekvencijom. *Android* sustav podržava frekvencije uzorkovanja 8000, 11025, 16000, 22050 i 44100 Hz. U razvijanoj aplikaciji korištena je frekvencija uzorkovanja od 11025 Hz. To teoretski znači da se pravilno može rekonstruirati ulazni signal frekvencijskog pojasa otprilike 0-5.5kHz, dok u praksi to ovisi o kvaliteti ulaza (mikrofon) i drugim parametrima.

3.3. Kvantizacija

Procesom uzorkovanja signal postaje diskretan u vremenu i kontinuiran po amplitudi (očitanje amplitudne vrijednosti u trenutku uzimanja uzorka). Da bi taj signal mogli obrađivati, potrebno je nakon uzorkovanja provesti proces kvantizacije kojim taj signal postaje diskretan po amplitudi. U principu, neograničen skup vrijednosti (kontinuiran) prilagođavamo i prikazujemo nekim od najbližih dostupnih (ograničen skup) vrijednosti pa kvantizaciju opisujemo kao proces aproksimacije kontinuiranog skupa vrijednosti. Jasna posljedica tog postupka je i gubitak određenih informacija ulaznog signala zbog zaokruživanja na najbliže dostupne vrijednosti.

Kvaliteta kvantizacije ovisi, osim o samom signalu, o rezoluciji – broju mogućih diskretnih vrijednosti (veličini ograničenog skupa dostupnih vrijednosti). Standardno se rezolucija definira brojem bitova: npr. 8 ili 16 bita, što su ujedno i dostupne rezolucije na *Android* uređajima. Korištenje rezolucije 8 bita značilo bi da postoji 2^8 mogućih vrijednosti na koje se amplitude zaokružuju. Povećanje rezolucije na 16 bita omogućilo bi korištenje čak 2^{16} amplitudnih vrijednosti i imalo bi za posljedicu mnogo bolje kvantiziranje signala.

Dostupne amplitudne vrijednosti mogu se definirati linearno ili nelinearno. Kod linearne kvantizacije sve diskretne amplitude su definirane s jednakim razmacima (npr. 256 mogućih amplituda se linearno dijele na jednako široke razine od 0 do 255). S druge strane, nelinearna kvantizacija uzima u obzir i karakteristike signala pa je dostupne razine moguće formirati ovisno o statistici signala. Pri tome se područje češće prisutnih amplituda preciznije kvantizira i koristi veći broj razina, dok područje manje zastupljenih amplituda sadrži manji broj razina – gubitak informacija tako je statistički niži. Nelinearna kvantizacija se standardno koristi kod digitalizacije govora s obzirom na nisku učestalost nižih amplituda u standardnom govoru.

3.4. Pulsno-kodna modulacija

Pulsno-kodna modulacija - PCM (eng. *Pulse Code Modulation*) - je standardna metoda koja se koristi da bi digitalno prikazali uzorkovan analogni signal (standardna metoda digitalizacije signala implementirana praktički u svom sklopovlju i sustavima s podrškom za dohvat i obradu signala - u našem slučaju zvuka).

Da bi signal mogli obrađivati u vremenskoj domeni bez upotrebe kompliciranih algoritama, treba baratati što jednostavnijom snimkom. PCM primijenjen na zvučni signal snimljen mikrofonom daje čistu digitalnu reprezentaciju analognog zvuka bez ikakve kompresije te time ubrzava proces obrade zvuka i omogućuje rad sa čistim uzorcima. Ako takvim zapisima dodamo standardima definirano zaglavlje, dobivamo standardnu, nekomprimiranu WAV datoteku. U zaglavlju glazbene datoteke obično se pohranjuju osnovne informacije o zvučnom zapisu kao što su naziv (*tag*), frekvencija uzorkovanja, veličina uzoraka i još mnoge druge. U razvijanoj aplikaciji nije implementirano korištenje WAV datoteka već se u analizi, pohrani i reprodukciji zvuka koristi čisti (eng. *raw*) PCM zapis, uz dodatno definirane konfiguracijske parametre u samom izvornom kodu (frekvencija uzorkovanja i ostali parametri potrebni za korištenje kasnije opisanih sistemskih biblioteka). Radi lakšeg razumijevanja PCM zapisa, na slici 2. je numerički prikaz signala – polje cjelobrojnih vrijednosti koje odgovaraju kvantiziranim vrijednostima amplituda u određenim trenucima.

```
+------+------+------+------+------+------+------+------
|  500 |  300 | -100 | -20  | -300 |  900 | -200 |  -50
+------+------+------+------+------+------+------+------
```

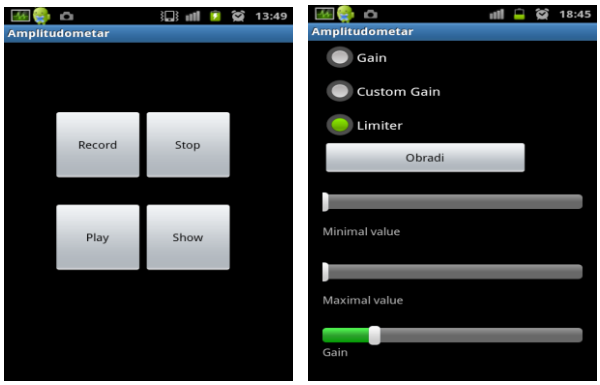
Slika 2. Polje s 8 uzoraka audiozapisa

Mada u ovom polju vidimo samo 8 uzoraka, svake sekunde je uzeto onoliko uzoraka kolika je frekvencija uzorkovanja, što bi značilo ako je frekvencija uzorkovanja 8000 Hz, svake sekunde je uzeto 8000 vrijednosti. Svaka od tih vrijednosti se opisuje s 8 ili 16 bita gdje isto vrijedi pravilo „više je bolje“. Također, na slici 2. se vidi da postoje i negativne vrijednosti. To je zbog same prirode zvučnog signala te njegovog utjecaja na mikrofon zbog kojeg se membrana mikrofona giba prema unutra i prema van. U razvijanoj aplikaciji podaci se spremaju u međuspremnik tipa *short*, što bi značilo da svaki uzorak može biti opisan sa 16 bita s obzirom na podršku za predznak broja, brojevima između -32767 i 32767.

Prema gore opisanim postavkama (uzorkovanje 8 kHz, kvantizacija 16 bita) količina podataka dobivenih u jednoj sekundi je 128000 bitova (128 kbit/s), što se dobije jednostavnim množenjem veličine svakog uzorka s frekvencijom uzorkovanja.

4. KORISNIČKO SUČELJE

S obzirom na primjenu, u razvoju aplikacije nastojalo se korištenje iste učiniti što jednostavnijim za krajnjeg korisnika. Sukladno tome definirano je jednostavno, ali funkcionalno korisničko sučelje. Na slici 3. prikazana su dva standardna sučelja razvijane aplikacije: Lijevo je početni ekran iz kojeg se pokreću sve glavne funkcije programa (gumbi za snimanje, preslušavanje, grafički prikaz i obradu zapisa). Na slici desno je sučelje za definiranje parametara i upravljanje ugrađenim funkcijama. Korišteni su standardni elementi grafičkog korisničkog sučelja – gumbi, izbornici i klizači. U sučelju za vizualizaciju signala korištena je *GraphView* knjižnica za izradu grafikona [4], dok je pozivanje funkcija za manipulaciju zapisima izvedeno korištenjem korisničkog izbornika.



Slika 3. Korisničko sučelje aplikacije

Android platforma omogućuje da grafičko sučelje bude opisno definirano standardnim XML jezikom, dok se funkcionalnosti implementiraju u programskom jeziku Java. Osnovna ideja je jednostavna - svaki element korisničkog sučelja definiran u XML datoteci potrebno je instancirati u programskog kodu (korištenjem identifikatora – atribut „*id*“) i potom povezati sa željenim funkcionalnostima kao što prikazuje sljedeći primjer:

XML datoteka

```
<Button
      android:id="@+id/start"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Record"  />
```

Java programski kod

```
Button start;

start = (Button)findViewById(R.id.start);
start.setOnClickListener(startRecOnClickListener);
```

Zajednički argument svim elementima korisničkog sučelja je „*id*“. Pomoću njega dajemo svakome objektu jedinstveno ime (identifikator) te nakon toga pristupamo programski. Svaki objekt korisničkog sučelja sadrži i određeno sučelje koje prati promjene stanja tog objekta. Primjer za definiranje funkcije koja se pokreće klikom na gumb „*Record*“ dan je u nastavku:

Java programski kod

```
OnClickListener startRecOnClickListener
= new OnClickListener(){
  public void onClick(View arg0) {

   Thread recordThread = new Thread(new Runnable()
   {
    public void run() {
              recording = true;
              startRecord();
    }
   });
   recordThread.start();
 }};
```

U danom primjeru implementirana je funkcija koja će kod pritiska na gumb „*Record*“ pokrenuti dretvu koja započinje snimanje te održava korisničko sučelje „budnim“.

5. SNIMANJE I REPRODUKCIJA ZVUKA

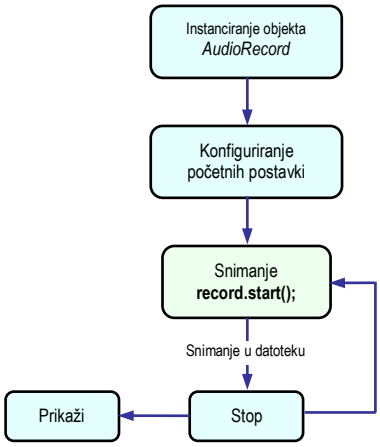
5.1. Snimanje zvuka

Zvuk se snima pomoću objekta *AudioRecord*. U primjeru programskog koda koji slijedi prikazan je konstruktor objekta *AudioRecord* te njegove početne postavke (frekvencija uzorkovanja 11025 Hz, 16 bit za kvantizaciju).

Java programski kod

```
AudioRecord audioRecord = new
AudioRecord(MediaRecorder.AudioSource.MIC,11025,
     AudioFormat.CHANNEL_CONFIGURATION_MONO,
     AudioFormat.ENCODING_PCM_16BIT,
     minBufferSize);
```

Da bi se objekt koristio, treba odrediti izvor zvuka, frekvenciju uzorkovanja, konfiguraciju kanala, veličinu svakog uzorka i međuspremnik u koji će se podaci privremeno spremati prije spremanja u datoteku. Dijagram koji prikazuje implementirani postupak snimanja prikazan je na slici 4.

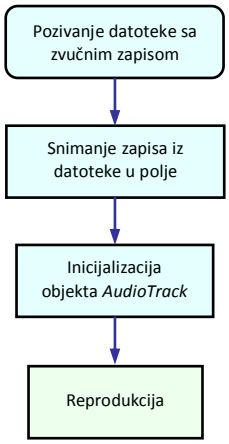


Slika 4. Dijagram toka - snimanje zvuka

Spremanje podataka se ne odvija direktno s mikrofona u datoteku, nego je potrebno instancirati međuspremnik (eng. *buffer*). Treba imati na umu da ako se želi nesmetano snimati s mikrofona na karticu moramo imati dovoljno veliki međuspremnik da ne bi došlo do preplavljivanja istog.

5.2. Reprodukcija zvučnog zapisa

Za reprodukciju snimljenog zapisa koristi se objekt *AudioTrack*. Tom objektu je kao parametre potrebno navesti frekvenciju kojom je snimljeni zapis uzorkovan i veličinu svakog uzorka – to je nužno da bi zvučni zapis bio pravilno reproduciran. Također, u konstruktor objekta *AudioTrack* potrebno je odrediti i način rada. Moguća su dva načina: *static* i *streaming*. Dok je *streaming* način rada za veće datoteke, *static* je za kratke uzorke zvuka. Postupak od pripreme zvučnog zapisa do njegove reprodukcije opisuje dijagram na slici 5.



Slika 5. Dijagram toka reprodukcije zvučnog zapisa

5.3. Prikaz snimljenog zapisa

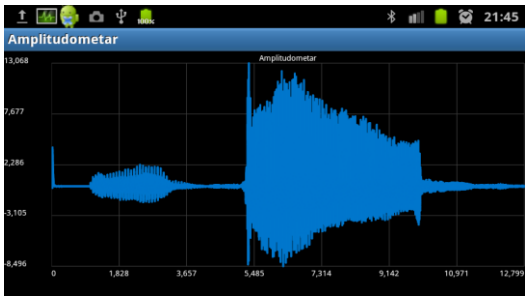
Vizualizacija snimljenog zapisa u vremenskoj domeni pokazala se složenim problemom u realizaciji. Da bi snimljeni zapis prikazali u obliku grafa, potrebna je poveća količina programskog koda za rezultate koji nisu vizualno primamljivi. S obzirom na to da postoje knjižnice koje nisu dio službenih Android API poziva, a koje omogućavaju jednostavnu izvedbu vizualizacije, odlučeno je da se u izradi aplikacije koristi jedna od takvih knjižnica. Takve knjižnice nazivaju se „*3-rd party library*“, tj. knjižnice razvijene od treće strane.

Za potrebe razvijenog alata koristila se knjižnica zvana *GraphView* [4]. Poziva se uz pomoć nekoliko vrlo jednostavnih naredbi, a isto tako se jednostavno popunjava podacima o objektu koji želimo vizualizirati. Primjer programskog koda u nastavku prikazuje pozivanje objekta *GraphView* u aplikaciji, te neke njegove mogućnosti.

Java programski kod

```
GraphViewData[] data = new
GraphViewData[audioData.length];

data[i] = new GraphViewData(i, audioData[i]);

GraphView graphView = new LineGraphView(this,
             "Amplitudometar");
graphView.addSeries(new GraphViewSeries(data));

graphView.setScrollable(true);
graphView.setScalable(true);
((LineGraphView) graphView).setDrawBackground(true);

LinearLayout layout = (LinearLayout)
findViewById(R.id.graph1);

layout.addView(graphView);
```

Primjer dobivenih rezultata vizualizacije (prikaz ekrana razvijane aplikacije u položenom (*landscape*) načinu rada) dan je na slici 6.



Slika 6. Vizualizacija zvučnog zapisa

6. IMPLEMENTIRANE FUNKCIJE ZA OBRADU ZVUKA

Implementirane funkcije za obradu zvuka dostupne su preko standardnih korisničkih izbornika. Operacije se provode nad aktualnim (vizualiziranim) zapisom, pri čemu po izvršenju operacije rezultat obrade postaje aktualan i grafički se prikazuje. Pomoćne funkcije za kontrolu uključuju mogućnost poništavanja provedenih obrada – povratak na originalni zapis kao i mogućnost preslušavanja aktualnog zapisa.

U studijskom primjeru – razvijenoj aplikaciji, implementirane su tri osnovne funkcije za manipulaciju zapisima u vremenskoj domeni. Svaka od funkcija je opisana u nastavku.

6.1. Limiter

Funkcija *Limiter* koristi se za ograničavanje amplitude snimljenog zapisa. Ako je amplituda uzorka veća od amplitude koju smo odredili kao maksimalnu, uzorak poprima vrijednost maksimalne određene amplitude. Sve funkcije nad signalom su izvršene jednokratnim prolaskom kroz analizirani zapis pomoću *while* petlje i *if* kontrole toka.

6.2. Custom Gain

Funkcija *Custom Gain* služi da bi se povećao samo određeni dio amplituda između minimalne i maksimalne vrijednosti određene u aplikaciju za vrijednost željenog pojačanja. Ako s tim pojačanjem vrijednost amplitude prelazi maksimalnu moguću vrijednost polja *short*, ona poprima maksimalnu vrijednost polja *short* da ne bi došlo do pojave zvane *clipping* koja pretvara koristan signal u šum. Da bi se moglo provjeriti je li nova vrijednost amplitude u granicama, potrebno je uvesti novu varijablu koja ima veći opseg od varijable tipa *short* kao npr. varijabla tipa *integer*.

6.3. Gain

Funkcija *Gain* je jednostavna funkcija pojačanja pomoću koje se cijeli snimljeni signal pojačava za određenu vrijednost. To se postiže tako da svaki dio polja u koje smo smjestili zvučni zapis pomnožimo s vrijednosti željenog pojačanja. Ponovno se koristi i varijabla pomoću koje kontroliramo eventualno prekoračenje maksimalne vrijednosti te po potrebi ograničavamo tu vrijednost.

7. ZAKLJUČAK

Operacijski sustav *Android* izrastao je od svog nastanka do danas u vrlo složen mobilni operacijski sustav koji ima pregršt opcija, a mogućnosti koje nema lako posudi. S obzirom na tržišnu zastupljenost, jasno je da sustav posjeduje velik potencijal za razvoj multimedijalnih aplikacija, čemu svjedoči količina njegovih *nativnih* API poziva kao i sama prilagodljivost uvjetima.

Ovim radom opisan je postupak implementacije jednostavne aplikacije za obradu zvuka na *Android* platformi. Dobiveni rezultati potvrdili su pretpostavke o mogućnostima sustava i dali smjernice za daljnji razvoj ovog ili sličnih rješenja. Jedna od interesantnijih ideja za daljnje nadogradnje jest implementacija frekvencijske analize zapisa i obrade u frekvencijskoj domeni (filtriranje i sl.). Također, interesantno je pitanje stvarne vremenske vizualizacije i analize signala na ulazu mikrofona. Činjenica je da uređaji trenutačno dostupni na tržištu svojim karakteristikama zadovoljavaju sve preduvjete potrebne za realizaciju i korištenje ciljane aplikacije. Očekivan napredak i povećanje performansi (procesorska snaga, radna memorija itd.) pametnih mobilnih uređaja stvara pretpostavke koje će nam u skoroj budućnosti omogućiti pokretanje sve složenijih aplikacija za digitalnu obradu zvuka.

8. LITERATURA

[1] Hižak D.: Razvoj jednostavnog alata za analizu zvuka na mobilnoj Android platformi, Završni rad br. 265/EL/2012, VELV Varaždin, 2012.

[2] http://developer.android.com

[3] http://www.eclipse.org – Eclipse razvojni sustav

[4] http://www.jjoe64.com/p/graphview-library.html - *GraphView* knjižnica

Kontakt autora:

mr.sc. Matija Mikac, dipl.ing.el.
Veleučilište u Varaždinu
J. Križanića 33, Varaždin
e-mail: matija.mikac@velv.hr

Dražen Hižak, bacc.ing.el.
e-mail: drazen.hizak@gmail.com